Multi-view Sequential Games: The Helper-Agent Problem

Christos Dimitrakakis
University of Lille, France
Chalmers university, Sweden
Harvard University, USA
christos.dimitrakakis@a3.epfl.ch

Firas Jarboui
ENSTA Paristech, France
ENIT, Tunisia
firasjarboui@gmail.com

David C. Parkes
Harvard University, USA
parkes@eecs.harvard.edu

Lior Seeman
Uber, Inc., USA
lseeman@cs.cornell.edu

ABSTRACT

Problems where agents wish to cooperate for a common goal, but disagree on their view of reality are frequent. Of particular interest are settings where one agent is an AI "helper agent" and the other is a human. The AI wants to help the human to complete a task but the AI and human may disagree about the world model. This may come about for example because of the limited rationality and biases of the human or because of misaligned reward models. In this paper, we formalize this as the multi-view sequential game, and show that even when the human's model is far from correct, an AI can still steer their behavior to more beneficial outcomes. In particular, we develop a number of algorithms, based on dynamic programming to discover helper policies for the AI, under different assumptions about the AI's knowledge. Experimentally, we show that the AI's beliefs about human model are not required to be accurate in order to act as a useful helper agent.

CCS Concepts

•**Human-centered computing** → *Collaborative interaction;* •**Computing methodologies** → **Multi-agent systems;**

Keywords

stochastic games, Human-AI co-operation, human-in-the-loop, Markov decision process, backwards induction

1. INTRODUCTION

Situated in the general area of value-aligned AI, we introduce a model of sequential multi-agent decision making for agents whose views of the world disagree. This can occur when agents have simply different utilities, beliefs, or computational bounds. It can lead agents to different conclusions about what the optimal joint policy is, even when they in principle agree on the objectives. Of main interest to us is the case where agents have the same utility function, but their models or utility estimates differ due to *bounded rationality or information.* This is particularly the case in scenarios where humans and artificially intelligent agents interact. The problem for the AI agent is to design a policy that takes into account human behaviour in order to help the human achieve its actual goal. This is the focus of the present paper.

As an example, consider humans trying to rescue people from a fire, with the aid of robotic agents. The robots may have accurate information about the layout of the building and so may be better equipped to make decisions about where to search. They should, however, take into account the limited knowledge of the human agents. Another example is a doctor trying to plan diagnostic tests for a patient. A diagnostic aid AI may provide a list of recommended tests for the doctor to perform, in order for the doctor to arrive at an accurate diagnosis with a small number of tests.

Formalising this setting leads to a class of sequential multi-agent decision problems, extending stochastic games. In particular, while in a stochastic game there is an underlying transition kernel to which all the agents agree, the same is not necessarily true in our setting. Each agent may have a different transition model. In particular, we assume that humans are incorrect about reality, and so the AI must plan taking this into account.

This framework that we develop is applicable to many scenarios involving human-AI collaboration, where there is agreement about the utility function, but disagreement about the optimal plan due to bounded human rationality. In this paper we model this bounded rationality through allowing the human to have an incorrect world model.

We focus on *collaborative policies*, where the AI takes actions directly in the environment together with the human. The human is assumed to observe the policy of the AI and to best respond to this policy, but for a potentially incorrect model of the world. We consider variations where in addition to the human having an incorrect model of the world, this (incorrect) model is not known by the AI.

In this paper, we examine the helper agent problem from the perspective of an AI trying to help a human. In particular, we model the problem as a generalized stochastic game, and study the power of the helper-AI intervention as the human's model of the world becomes more noisy and also considering that the AI may not completely understand the human's viewpoint. We investigate computational solutions

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

to this problem and demonstrate that the AI can indeed be helpful, even when it misunderstands humans.

In particular, we show that these types of games are significantly different from standard sequential games, as they are not efficiently solveable via backwards induction, even when players take alternating moves with full information and with a finite time horizon. In this paper, we study the effect of a dynamic programming algorithm for finding a policy for the AI, when the human's model is known. In particular, we develop a backwards induction algorithm, which maintains separate transition matrices and values for both players, to find approximately optimal policies for the AI. We provide a number of experimental results for applying this framework to the problem of AI-helper agents. We show that the helper-AI will improve the human's performance unless the human's model is extremely noisy, and as long as the AI understands the human's world model well enough. However, blithely assuming that human model is correct and identical to the planner's can lead to a moderate performance loss. We tested our algorithms on three types of environments. Firstly, hand-crafted environments with small numbers of states, designed to maximise the discrepancy between the views of the human and AI. Secondly, randomly generated environments. Finally, an environment corresponding to a problem in sequential experiment design.

1.1 Related work

A related problem is how to use internal rewards to improve the performance of computationally-bounded, reinforcement learning agents [13]. For example, even a myopic agent can maximize expected utility over a long time horizon if augmented with appropriately designed internal rewards. This model can be cast into our framework by allowing the helper agent actions to directly provide rewards to the human. However, unlike AI, it is less natural to consider a reward deliver from the robot to the person. Consequently, we focus on the AI taking actions in the same environment as the human.

Another related problem is how people can be used to train an AI agent [2]. This essentially allows the teacher to demonstrate how to perform a given task. The problem is how to make the best use of opportunities for demonstration. This setting naturally falls within our framework, where the teacher takes the role of our AI and the learner the role of our human. We would need to specify an appropriate advice-giving set of actions for the teacher agent. But previous work assumes that agents have a consistent environment model. More precisely, if the learner plays action a at state s, and the teacher's optimal policy plays a', that doesn't necessarily mean that the learner's assumed optimal continuation matches that of the teacher. Consequently, the learner may perform even worse under if he changes his actions.

The work most closely related to ours is that of *environment design* [15, 14] and *cooperative inverse reinforcement learning* [6]. In these problems, an AI observes a human agent performing a task, and uses inverse reinforcement learning [10] to estimate the utility function of the human. In environment design, the AI is assumed to associated a limited supply of rewards with different states, and makes multiple interventions in order to modulate the policy selected by the person. In cooperative iRL, the problem is that the AI wants to co-operate with the human in a task, but does not know what the task is. In particular [6] applies this framework to a set of two-stage game problems where the human demonstrates in the first stage and the AI imitates in the second stage. In particular, they show that the human must take into account the AI's best response when providing demonstrations and they develop a feature-matching algorithm for computing an appropriate demonstration policy for the human. Experimentally, they show that this has a significantly improved performance to uninformed demonstrations. However, it is not at all clear how the human would be able to compute or implement this demonstration policy. The focus of our work is entirely different. Firstly, in our formal model, we do not assume the human and AI have the same environment model. Furthermore our goal is not to infer the rewards of the human agent, but rather to compute a plan so as to optimally guide a human agent with an incorrect model.

Our work also relates to methods for computing optimal strategies in stochastic games [3, 16], but differs in that the agents (here an AI and a human) have different models of the transitions between states. Gal and Pfeffer [5] have also introduced a general formalism for representing agent beliefs, which could be used to model non-aligned beliefs. However, our focus is specifically on the dynamic, cooperative (multi-view) setting and the question of how to solve the AI's planning problem.

Finally, our methodology for computing AI policies when there is uncertainty about the human's model builds upon the minimax methods developed in [8] for Markov decision processes with uncertain transition probabilities.

2. THE MULTI-VIEW SEQUENTIAL GAME

We model a setting with two players, where we refer to the first player as the AI and the second player as the human. The agents act in a shared world, with the AI and then the human taking an action in each state (the human having observed the action of the AI), leading to rewards and state transitions. Thus, the state dynamic is defined on action pairs— an action of agent 1, and one of agent 2. The way the game is structured is that the AI selects a policy, this policy observed by the human, who then selects a policy.

Each player holds a specific model for the transition law of the dynamic environment, and the players may not agree on what the law is. We assume the AI has the correct world model, and thus when there is disagreement it is the human's model that is incorrect. The AI also forms a belief about the model of the human (either a point belief, or a set-based belief.) We also allow disagreement about the reward function, and in general we allow neither reward function to be correct. To keep the set-up simple, and because we prefer to focus in this first paper on issues involving disagreement in the model rather than reward, we will assume here that if there is disagreement the AI has the correct knowledge of the reward function of the human.

Definition 1. (*Multi-view stochastic game*) A multi-view stochastic game μ is a 2-player game with $\mu = \langle \mathcal{S}, \mathcal{A}, \rho, \beta, B \rangle$, where

- $\mathcal{S}$ is a state space.
- $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$.
- $\rho = (\rho_1, \rho_2)$ is a pair of reward functions $\rho_i : \mathcal{S} \to \mathbb{R}$.

- $\beta = \{\beta_1, \beta_2\}$ is a set of probability laws, which represent each agent's world model.
- B is the belief agent 1 has about agent's 2 model. (B is either a point model, or a set of possible models of agent 2).
- If $\rho_1 \neq \rho_2$, then agent 1 knows the reward function of agent 2. Moreover, in this case we also define ρ^*, which we view as the correct reward function for the human.

At time t, both players observe the state s_t, and player 1 selects an action $a_{t,1} \in A_1(s_t) \subseteq \mathcal{A}_1$. Player 2 observes $a_{t,1}$ and selects an action $a_{t,2} \in A_2(s_t, a_{t,1}) \subseteq \mathcal{A}_2$. The joint action is denoted by $\mathbf{a}_t = (a_{t,1}, a_{t,2})$.

The model of an agent about the world is encapsulated by the probability law β_i, such that:

$$s_{t+1} \mid s^t, \mathbf{a}^t \sim \beta_i(s_{t+1} \mid s^t, \mathbf{a}^t), \tag{2.1}$$

where $s^t = s_1, \ldots, s_t$ and $\mathbf{a}^t = \mathbf{a}_1, \ldots, \mathbf{a}_t$. The AI agent also has a belief B about the human agent's model. In particular, agent 1 has a set $B = \left\{ \beta_2^{(k)} \;\middle|\; k \right\}$ of probability laws, each one of which represents a possible model for the second agent.

Each agent is interested in maximising utility at t with respect to its own reward:

$$u_{i,t}(s^T) \triangleq \sum_{k=t}^{T} \gamma^{t-1} \rho_i(s_k), \tag{2.2}$$

given a sequence of future states s^t. We also use $u_i \triangleq u_{i,1}$ to denote the utility starting from the first step.

For the rest of the paper, unless otherwise state, we assume that $\rho^* = \rho_1$. We also make the following, stronger assumption.

Assumption 1. *Agent 1's model is correct.*

This assumption is natural in our setting, where the AI is helping the human in decision making and especially helping the human to overcome some bounded reasoning patterns or mistaken models. Hence, the actual world model used in all of our experimental results will correspond to Agent 1's transition model. Because of this assumption, then if in addition the AI's reward function is correct in that it reflects what the human's reward "should be" (perhaps distinct from the reward function ρ_2 adopted by the human), then the AI's expected utility reflects that of a benevolent and correct planner.

In most of the specific problems that we consider, we further assume $\rho_1 = \rho_2$, and assume this is the correct reward function for the human. The only exception is the *experiment design* problem (Example 1). There the reward implicitly depends on an agent's model (since it is induced from information states), and so the utilities of the two agents differ when their world models differ. In this particular scenario, we assume that Agent 1 knows Agent 2's reward (this follows from agent 1 knowing agent 2's world model.)

Assumption 2. *All players in this game have Markov policies. The first player's policy being $\pi_i(a_{t,1} \mid s_t)$ and the second player's being $\pi_i(a_{t,2} \mid s_t, a_{t,1})$. We denote joint policies by $\pi = (\pi_1, \pi_2)$.*

This second assumption is w.l.o.g. when the state is the complete observation history.

2.1 Properties

While in Markov decision processes (MDPs) it's possible to find policies that are optimal for any starting state, this is not the case here, as we shall see later. For this reason we consider the following solution concept.

Definition 2. (*Optimal policy*) Let σ be a distribution over starting states s_1. Then a joint policy π is optimal under σ, β_1 iff

$$u(\pi) \triangleq \mathbb{E}^{\pi}_{\beta_1,\sigma}\, u_1 \geq \mathbb{E}^{\pi'}_{\beta_1,\sigma}\, u_1 \qquad \forall \pi',$$

where

$$\mathbb{E}^{\pi}_{\beta_1,\sigma}\, u_1 = \sum_{t=1}^{T} \gamma^{t-1} \sum_{s \in \mathcal{S}} \rho^*(s) \beta_1(s_t = s \mid \pi, \sigma).$$

The optimal policy is defined under the model of agent 1, which we assume to be the correct world model. The reward function adopted is the optimal reward function, and may differ from ρ_1 or ρ_2.

If player 1 fixes their policy, this creates an MDP for player 2, who can then easily compute their best response policy. In our setting, player 1 announces its policy to player 2. Then player 1 can take into account the reaction of player 2 to his policy. In Markov decision processes, it is traditional to use the concept of a value function. This is the expected utility of a policy from a given starting state. The same concept can be used in our setting. More precisely, for a given decision process μ, the value function V, defined as:

$$V^{\pi}_{i,t}(s) \triangleq \mathbb{E}^{\pi}_{\beta_i}(u_{i,t} \mid s_t = s) \tag{2.3}$$

is the expected utility according to i's model, if we start from state s and follow joint policy π. This policy can be found with a number of dynamic programming algorithms such as backwards induction (for the finite horizon case) and the numerous variants of policy and value iteration for the infinite horizon case [11].

In MDPs there is always a uniformly optimal policy that satisfies $V^{\pi}_t(s) \geq V^{\pi'}_t(s)\ \forall s$. However, in our setting this is not necessarily the case. When the first player selects a policy, they other player follows by selecting a response. The analogy in the MDP setting would be that the dynamics of the problem change depending on the policy. Thus, it is entirely possible that there is no uniformly optimal policy for the first player. This can be seen by the following counterexample.

Example 1: Non-uniform domination

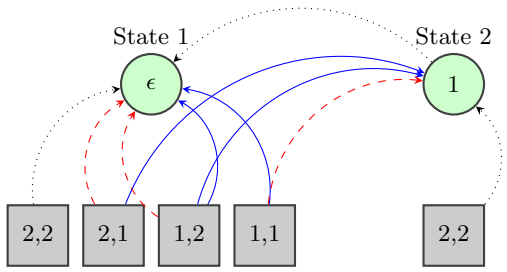


Figure 1: Non-dominating counter-example. States are indicated by circles, with their reward written within. Squares indicate action pairs at each state. Black dotted lines indicate agreement in the model, while solid blue lines the model of player 1, and red dashed ones that of player 2. Omitted action pairs have arcs directly from the state.

In this example, the players agree on all rewards. They also agree on all actions in state 2, and about the action (2,2) in state 1. However, they think (1,1) and (2,1) have opposite effects in state 1. They only partially agree about (1,2), which player 1 thinks has only 2/3 probability of staying in state 1.

Unlike MDPs, in Example 1, there is no optimal stationary policy for the infinite horizon discounted case. Moreover, there is a non-stationary policy that outperforms all stationary policies. These policies are generally cyclic, i.e. they obey $\pi(\mathbf{a}_{t+k} \mid s_{t+k}) = \pi(\mathbf{a}_t \mid s_t)$ for some $k > 1$. It is, however, possible to determine them using dynamic programming. In fact, [16] describes a backwards induction method for discovering cyclic equilibria in Markov games.

π_1	π_2	V_1		V_2	
1,1	1,1	2	2.9	10.77	11
1,2	1,2	2	20	19.1	20
2,1	1,1	10.77	11.23	2	2.9
2,2	2,2	2	20	2	20
cyclic	cyclic	19.1	20	16.6	20

Table 1: π_i are deterministic policies, taking actions a_1, a_2, a_3 in each respective state. The i-th column of V_j shows $V_j(i)$. It can be seen that player 1 has no dominating policy.

Table 1 shows the state values of different policies of the two players for this example, when player 2 plays the optimal response to player 1.[1] There is no stationary policy for player 1 that is optimal from all starting states. However, there is a cyclic policy that achieves this.

Finally, there are also problems which induce a cyclic behaviour, even if there is a uniformly optimal policy for them. Example 2 shows a particularly simple such problem.

[1]See Section 3 for a precise definition.

Example 2: Cyclic

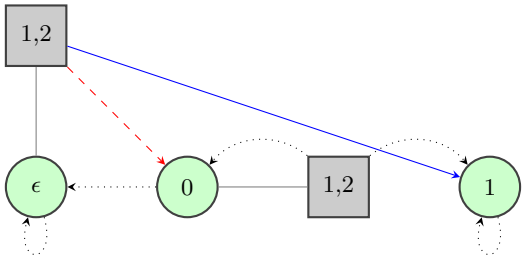


Figure 2: States are indicated by circles, with the reward at each state written within. Squares indicate action pairs at each state. Black dotted lines indicate agreement in the transition, while solid blue lines the model of player 1, and red dashed ones that of player 2.

IN this example, the players agree on all rewards. There are three states, and two actions for each player. For the left-most state, the two agents disagree on the effect of pair (1,2), with the first player's model corresponding to the solid blue arrows and the second player's to the dashed red arrows. The agents agree everywhere else. In the middle state, the action pair (1,2) remains there and leads to the rightmost state with equal probability.

3. ALGORITHMS

When player 1 fixes a policy π_1, player 2 needs to solve a maximisation problem. From player 2's perspective, this is an MDP, and so a uniformly optimal policy can be found. Thus, there is always a well-defined best-response policy:

$$\pi_2^*(\beta_2, \pi_1) = \arg\max_{\pi_2} \mathbb{E}_{\sigma,\beta_2}[u_2 \mid \pi_1, \pi_2]. \tag{3.1}$$

Consequently, given that player 2 computes a β_2-best response, player 1 can always compute the value of any policy π_1:

$$\mathbb{E}_{\sigma,\beta_1}[u_1 \mid \pi_1, \beta_2] = \mathbb{E}_{\sigma,\beta_1}[u_1 \mid \pi_1, \pi_2^*(\beta_2, \pi_1)]. \tag{3.2}$$

Then player 1 needs to find a policy

$$\pi_1^*(\beta_1, \beta_2) \in \arg\max_{\pi_1} \mathbb{E}_{\sigma,\beta_1}[u_1 \mid \pi_1, \beta_2]. \tag{3.3}$$

In the case where player 1 has the correct reward, and the correct world model, then this will is optimizing the expected utility created by the joint policy induced by the two players. Unfortunately this is not a zero-sum problem. This can be seen by writing it in standard form:

LEMMA 1. *A multiview stochastic game μ can be written as a game $G_q(\mu) = \langle \pi, v \rangle$ with joint strategies π and values v, such as the value of the k-th policy tuple $\pi^{(k)} = (\pi_1^{(k)}, \pi_2^{(k)})$ for player i is:*

$$v_{k,i} = \mathbb{E}_{\sigma(s),\beta_i}[u_i \mid \pi_1^{(k)}, \pi_2^{(k)}]. \tag{3.4}$$

a subgame perfect Nash equilibrium must exist, the policy space is rather large. For example, for infinite horizon undiscounted problems, we must at least consider all stationary policies, which are $O(|\mathcal{S}|^{|\mathcal{A}|})$. For that reason, we develop an approximate algorithm based on dynamic programming.

3.1 Dynamic programming approximation

We now describe a dynamic programming approximation. First, we show how to formulate it as a backwards induction algorithm for the case where the AI knows the model of the human. We remind the reader that that player 1 communicates its policy π_1 to player 2 and that player 2 can see player 1's action before he chooses his own. At stage t of the program, player 2 has observed the current state s_t and the action $a_{t,1}$ of the first player, and also knows the future policy of player 1. He now chooses his action

$$a^*_{t,2}(a_1) \in \arg\max_{a_2} \rho(s_t) + \gamma \sum_{s_{t+1}} \beta_2(s_{t+1}|s_t, a_1, a_2) V_{2,t+1}(s_{t+1}).$$

For every state and player-1-action, there is a well-defined continuation for player 2. Now, player 1 needs to define his action. This can be done easily, since we know player 2's continuation, and so we can define a value for each state-action pair for player 1:

$$Q_{1,t}(s_t, a_{t,1}) = \rho(s) + \gamma \sum_{s_{t+1}} \beta_1(s_{t+1}|s_t, a_{t,1}, a^*_2(a_{t,1})) V_{1,t+1}(s_{t+1}).$$

Action $a^*_{t,1}$ can now be defined as maximizing $Q_{2,1}$. What remains is to define the values of the current state for both players, which is

$$V_{i,t}(s_t) = Q_{i,t}(s_t, a^*_{t,i}). \tag{3.5}$$

Optimality.

Contrary to the MDP setting, this algorithm may not obtain the optimal policy. In addition, for the discounted infinite horizon case, it may not converge to a stationary policy. This is because of the dependency between the future policy of player 1 affects the current action of player 2, and so the effective transition matrix for player 1. More precisely, the transition actually depends on the future joint policy $\pi^{n+1:T}$, because this determines the value $Q_{2,t}$ and so the policy of the second player. In practice, this results in a cyclic behavior for infinite-horizon discounted games. Similar cycles have been observed before in non-zero sum Markov games [16]. To handle cycles for infinite-horizon problems in the experiments, we simply test each of the instantaneous policies π_1 in the cycle, find the best response of player 2, and then select the best policy among those.

To assess the optimality of the algorithm, we compared it with a simple local search algorithm[2] for stochastic policies. The table below shows a sample of results for a number of environments. The first method is the dynamic programming algorithm described. As in Section 2, we compared this against the stationary policies extracted from one of the cycles of the algorithm. We also considered stationary stochastic policies found by local search, which could frequently improve over the dynamic program.

It is also interesting to note that if we assume that the human's model and reward matches that of the AI, the problem reduces to an MDP. From result on approximate MDPs [4], we know that our utility loss will be bounded by the L1 distances between our assumption and reality. In particular, for a finite-horizon γ-discounted problem, the error will be $O(\|\beta_1 - \beta_2\|_1 (1-\gamma)^{-2})$.

[2]More precisely, a line search algorithm on the principal axes, combined with random restarts.

Cyclic	11	12.8	10.4	0.9	8.5	19.6
Stationary	11	2.8	9.0	0.9	8.4	11
Local	11	13.4	10.5	0.9	9	19.4

Table 2: Expected utility of policies found by three different algorithms in 6 different environments.

3.2 The case of unknown beliefs.

In this setting, we assume that we just know that $\beta_2 \in B$. In practice, the AI can simply construct a ball B around its own model, or sample a set of possible model from some distribution. In either case, the question of which belief to plan against is a problem. In prior work in MDPs with uncertain transitions, both minimax or maximax choices were considered [8]. We shall consider only the minimax problem here, as we believe it may provide robustness. Even then, though, we have two possible solutions. The first is to consider the minimax problem:

$$\min_{\beta_2 \in B} \max_{\pi_1} \mathbb{E}_{\sigma,\beta_1}[u_1 \mid \pi_1, \pi^*_2(\beta_2, \pi_1)], \tag{3.6}$$

$$\pi^*_2(\beta_2, \pi_1) \in \arg\max_{\pi_2} \mathbb{E}_{\sigma,\beta_2}(u_2 \mid \pi_1, \pi_2). \tag{3.7}$$

For a finite B, this simply requires enumerating all possible models. After finding the minimising β_2, we simply play the best response π_1 to it. Unfortunately, this is somewhat optimistic for our purposes, and it is probably more robust to solve the corresponding maximin problem.

$$\max_{\pi_1} \min_{\beta_2 \in B} \mathbb{E}_{\sigma,\beta_1}[u \mid \pi_1, \pi^*_2(\beta_2, \pi_1)] \tag{3.8}$$

The maximin problem is significantly harder, as it requires enumerating all policies π_1. However, when B is a factored space, so that each part of the human's transition model can be changed independently of the others, we can use the following DP approach to find a maximin policy:

$$\hat{\beta}_t \in \arg\min_{\beta_1} \mathbb{E}_{\beta_1}[V^*(s_{t+1}) \mid s_t, a_{t,1}, a^*_{t,2}] \tag{3.9}$$

$$Q_{1,t}(s_t, a_{t,1}) = \rho_(s_t) + \gamma \, \mathbb{E}_{\beta_2}[V^*(s_{t+1}) \mid s_t, a_{t,1}, a^*_{t,2}] \tag{3.10}$$

$$a^*_{1,1} \in \arg\max_{a_1} Q_{1,t}(s_t, a_1) \tag{3.11}$$

$$V_{i,t}(s_t) = Q_{i,}t(s_t, a^*_{t,1}) \tag{3.12}$$

In particular, if B is composed of a set of polytopes describing sets of transition probabilities for each state-action pair, then we only need to consider the vertices of this polytope. This is the case when B is the set of transition distributions within an L1 ball of the AI's model, i.e. when B is the set of models β_2 satisfying $\|\beta_1(\cdot \mid s, a) - \beta_2(\cdot \mid s, a)\|_1 \leq d_{s,a}$ for some permissible error bounds $d_{s,a}$. This is essentially the approach used in the optimistic value iteration algorithm for UCRL2 [7].

Example 1: Experiment design

A classical goal in experiment design [12, 9] is to maximise the expected information gain after T experiments. We assume the AI has an accurate probability table of different parameters, observations and experiments, while the human's model maybe incorrect. It is instructive to consider the particular case where the AI suggests a set of experiments, and the human selects one of them.

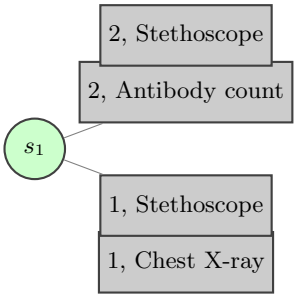


Figure 4: In this illustration, the AI selects a set of exams out of which the physician can choose. The AI has two actions, the first leaves the human with a choice between using a stethoscope or a blood exam for antibody count. The second gives a choice between a stethoscope and an X-ray.

In this setting there is a true parameter θ^* (the disease status) and we can make experiments a, observing outcomes x. Hence the state at time t is the complete history of experiments and outcomes $s_t = (a_k, x_k)_{k=1}^{t-1}$, with s_1 being empty. This can be seen as an information-state MDP, where each agent i has a prior belief $\xi_i(\theta)$ and a set of conditional probabilities $p_i(x|a, \theta)$ over all outcomes, given an action $a = (a_1, a_2)$, and disease status θ. Then the transition distribution for each player is decomposed as:

$$\beta_i(s_{t+1} = (s_t, a_t, x) \mid s_t, a_t) = \sum_{\theta \in \Theta} p_i(x \mid a_t, \theta) \xi_i(\theta \mid s_t),$$

where is $\beta_i(\theta \mid s_t)$, is the posterior distribution arising from the history s_t and the likelihood model ξ_i. After T tests, the game ends and the players obtain reward

$$\rho_i(s_T) = \sum_{\theta \in \Theta} \xi_i(\theta \mid s_T) \ln \xi_i(\theta \mid s_T),$$

equal to the negative entropy of the parameter posterior, while $\rho_i(s_t) = 0$ for $t < T$. Note that in this scenario, the human's expected reward will not match the AI's due to the different posterior.

4. EXPERIMENTS.

We set out to investigate the performance of a human agent acting in an environment with possibly incorrect beliefs. We model this by the human using a transition matrix that deviates from the true world model. In all cases, we measure the expected utility with respect to correct world model, and provide the AI with the correct world model. Apart from the experiment design setting, both players rewards agree.

4.1 Environments and simulation

We experimented on the two detailed domains given in Examples 1 and 2, a number of randomly generated infinite-horizon problems, as well as randomly generated experiment design problems with the structure described in Example 1. In all cases, we measured performance in terms of the AI's model of the world, i.e. we assumed its correctness. To model human inaccuracy, the human model was a noisy version of the AI's model. All simulations were run 100 times for each measured point.

Infinite-horizon problems.

For those problems, we set $\gamma = 0.95$. The randomly generated domains had up to 16 states and 4 actions per agent. They were created by sampling transition matrices from uniform distributions on the simplex. All environments had reward of 0.1 in one state, and 1 in another state, with 0 everywhere else.

The human model for these problems was drawn from an product-Dirichlet prior that was $1/\epsilon$-concentrated around the AI's belief.

Experiment design.

We tested randomly generated experiment design problems with 2 conditions, observations and actions per agent, and with a planning horizon of up to 3 steps. Both the AI and human had the same initial belief over the condition. However, the human's likelihood model p_2 was generated by ϵ-contaminating the AI's model with the uniform distribution. This avoided creating loops in the information-state MDP for the human (though the AI has no way of knowing that).

4.2 Aiding an incorrect human

In our experiments, we considered four kinds of joint policies. We wished to test whether the AI could help improve the joint policy, even when it did not have an accurate idea of the human's beliefs. We examined the behaviour of a number of joint policies as the human's accuracy degrades with increasing ϵ. We tested each environment for different values of ϵ, with a different generated human belief each time, and measured the average over 100 trials. The cases we considered are given below.

Human only.

The human selects a joint policy according to its belief. This corresponds to the human programming the AI with the policy he thinks would be optimal and the AI and the human then acting in the world.

AI Known.

In this case, the AI is not only autonomous, but it also knows the belief of the human. We expect that in that case, human performance can be significantly improved even when human beliefs are far from being correct.

AI Unknown.

In this case, the AI does not know the human's belief.

More specifically, the AI assumes that its beliefs and those of the human are the same. While this may still help, in some cases it has the opposite effect.

AI Sample.

Finally, we also considered the case where the AI assumes a set of possible beliefs for the human, different from its own. In that case, decision making for the AI involves also solving a minimax problem.

4.3 Results

The experimental results for the above cases are shown in Figure 3. The horizontal axis is the amount of error of the human model, while the vertical axis is the expected utility measured under the model of the AI. We show results both for the infinite horizon domains (a-g) and for the experiment design domain (h).

We can see that when the human is correct, all cases obtain the same performance. However, the human's utility quickly deteriorates when he makes all the decisions by himself. If the AI knows exactly what the human model is, then it can significantly increase his performance. However, it is also interesting to investigate what happens when this is no longer true. We see that when the AI simply guesses that the human has the same model as it does, the joint policy is sometimes even worse than only when the human takes decisions. Perhaps surprisingly, sampling and using the minimax policy does not improve upon simply assuming that the human's model is the same as the AI's most of the time (see 3(h)). This might be due to the fact that these policies are overly pessimistic.[3]

5. CONCLUSION AND FUTURE WORK

We introduced a framework for AI-helper agents. This relies on the premise that an AI will be useful for tasks where it could perform *better* than humans, due to its superior knowledge or computational capabilities. For this reason, we assume that the AI and human have different models of the world, and perhaps different utilities, but that the AI is correct. We show that games of this type, while not solveable exactly by efficient algorithms, can still be tackled with a simple dynamic programming method. Our experiments over a large number of domains, including a realistic experiment design problem, demonstrate that as the human's internal model deteriorates, the AI can still help him maintain his performance close to the best possible level. This was possible even when the human's model was unknown, and the AI simply assumed that he had an identical model to its own.

In future work, we would like to examine algorithms that scale better computationally and that do not suffer from the problems of the dynamic programming heuristic. One idea is to extend gradient methods developed in [1] to the sequential setting. This might be particularly interesting since our results in Table 2 indicate that some performance gain can be achieved using stochastic policies when the human model is known. However, it is unclear whether there exists a viable method for the case when it is unknown.

We would also like to explore different assumptions for this problem. In particular, it would be interesting to see how to deal with the case where we assume that the human does not know the AI's policy, and only observes its actions. Finally, a natural extension of our work is to consider the case where the AI learns the reward or the world model of the human agent.

[3]Minimax results on the other domains not shown to avoid clutter.

REFERENCES

[1] K. Amin, S. Singh, and M. Wellman. Gradient methods for stackelberg security games. In *UAI 2016*, 2016.

[2] O. Amir, E. Kamar, A. Kolobov, and B. Grosz. Interactive teaching strategies for agent training. In *IJCAI 2016*, 2016.

[3] B. Bošanský, S. Brânzei, K. A. Hansen, P. B. Miltersen, and T. B. Sørensen. *Computation of Stackelberg Equilibria of Finite Sequential Games*, pages 201–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[4] E. Even-Dar and Y. Mansour. Approximate equivalence of markov decision processes. In *Learning Theory and Kernel Machines. COLT/Kernel 2003*, Lecture notes in Computer science, pages 581–594, Washington, DC, USA, 2003. Springer.

[5] Y. Gal and A. Pfeffer. Networks of influence diagrams: A formalism for representing agentsâĂŹ beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, 33(1):109–147, 2008.

[6] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell. Cooperative inverse reinforcement learning, 2016.

[7] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

[8] R. E. L. Jay K. Satia. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.

[9] D. V. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27(4):986–105, 1956.

[10] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[11] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. 1994.

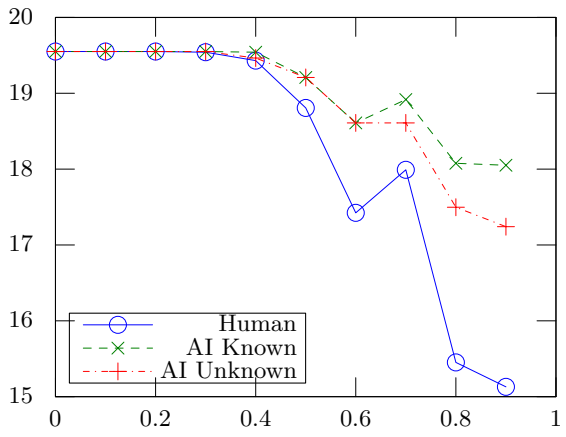
[12] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[13] J. Sorg, S. P. Singh, and R. L. Lewis. Internal rewards mitigate agent boundedness. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1007–1014, 2010.

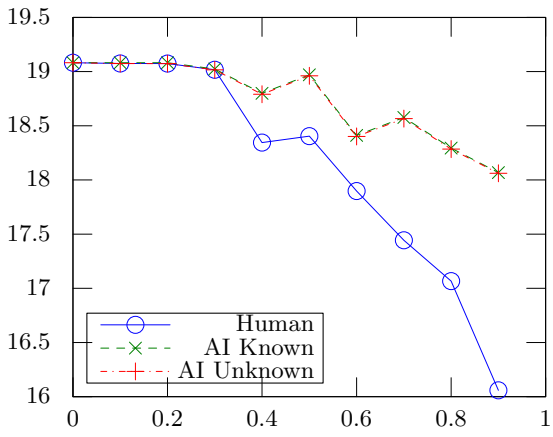
[14] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, page 208–214, Chicago, IL, July 2008.

[15] H. Zhang, D. C. Parkes, and Y. Chen. Policy teaching through reward function learning. In *10th ACM Electronic Commerce Conference (EC'09)*, page 295–304, 2009.

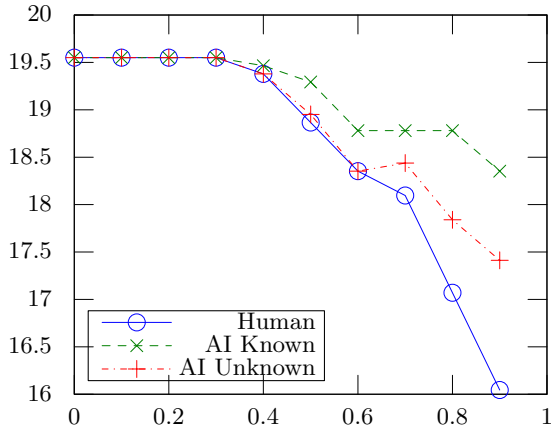
[16] M. Zinkevich, A. Greenwald, and M. Littman. Cyclic equilibria in markov games. In *Advances in Neural Information Processing Systems*, 2006.



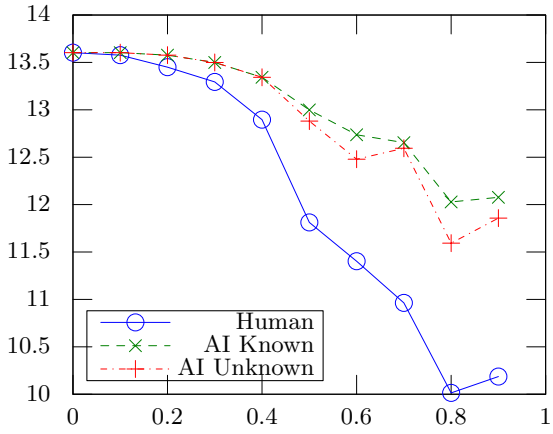
(a) Non-uniform domination



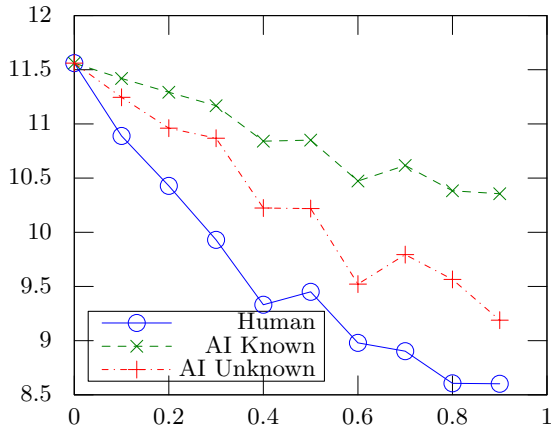
(b) Cyclic



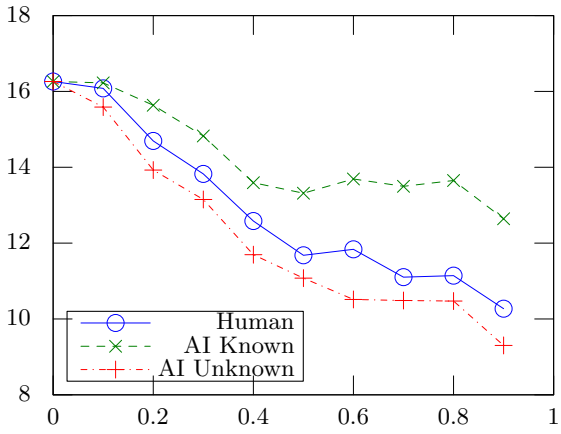
(c) Deterministic



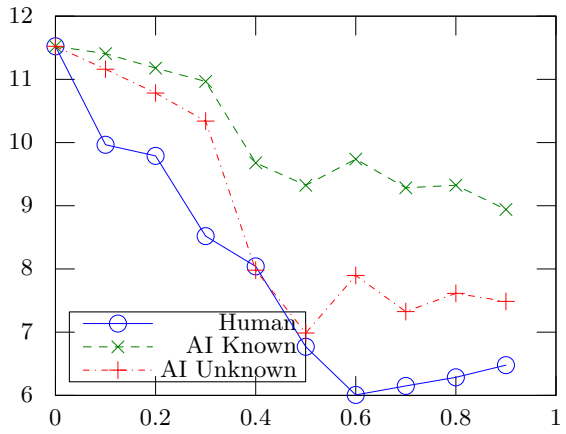
(d) Random $|\mathcal{S}| = 2, |\mathcal{A}_i| = 2$



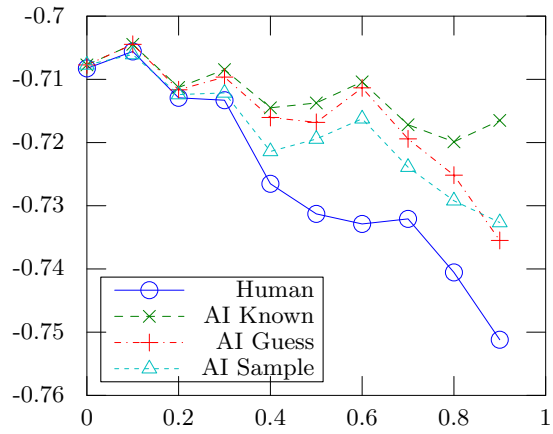
(e) Random $|\mathcal{S}| = 3, |\mathcal{A}_i| = 2$



(f) Random $|\mathcal{S}| = 4, |\mathcal{A}_i| = 2$



(g) Random $|\mathcal{S}| = 5, |\mathcal{A}_i| = 4$



(h) Experiment design, $|\mathcal{A}_i| = 2, T = 3$

Figure 3: Effect of incorrect human knowledge. The horizontal axis denotes the deviation of the human's model from reality. The vertical axis is the expected utility of the policies described in Section 4.2.